## KROK 1: Inicjalizacja repozytorium i struktura folderów

mkdir my\_app && cd my\_app  
git init

Utwórz plik .gitignore i zapsiz w nim:

node\_modules/

Struktura katalogów:

my\_app/  
├── backend/  
│ ├── main.py  
│ └── requirements.txt  
├── electron/  
│ └── package.json  
├── frontend/  
│ ├── index.html  
│ ├── vite.config.js  
│ ├── package.json  
│ └── src/  
│ ├── main.jsx  
│ ├── App.jsx  
│ └── index.css  
├── main.js (główny plik Electron)  
├── package.json (główny plik projektu)

## KROK 2: Instalacja Electron

npm init -y  
npm install electron --save-dev

**Cel:** Instalujemy Electron globalnie dla całego projektu (w katalogu głównym).

## KROK 3: FRONTEND (React + Tailwind)

npm create vite@latest frontend -- --template react -y  
cd frontend  
npm install  
npm install -D tailwindcss @tailwindcss/vite

### frontend/vite.config.js

import { defineConfig } from 'vite';  
import react from '@vitejs/plugin-react';  
import tailwindcss from '@tailwindcss/vite';  
  
export default defineConfig({  
 base: './',  
 plugins: [react(), tailwindcss()]  
});

**Cel:** Włącza obsługę Tailwind jako plugin w Vite

### frontend/src/index.css

Zamień zawartość na:

@import "tailwindcss";

### frontend/package.json

W pliku frontend/package.json dodaj pole homepage, aby poprawnie działał z Electronem. Umieść je obok pól takich jak name i version:

{  
 "name": "frontend",  
 "version": "0.0.0",  
 "homepage": "./",  
 ...  
}

### Budowanie Reacta

npm run build

**Cel:** Tworzy statyczne pliki Vite (frontend/dist) dla Electron

## KROK 4: BACKEND (FastAPI)

Utwórz środowisko conda:

conda create -n myapi python=3.10  
conda activate myapi

Zainstaluj potrzebne biblioteki:

pip install fastapi uvicorn  
pip freeze > backend/requirements.txt

### backend/requirements.txt

Plik potrzebny tylko w trakcie tworzenia aplikacji. Używany przez kolejnych deweloperów do szybkiej instalacji zależności w środowisku Pythona (np. conda lub venv) za pomocą pip.

fastapi  
uvicorn

### backend/main.py

import uvicorn  
from fastapi import FastAPI  
from fastapi.middleware.cors import CORSMiddleware  
  
app = FastAPI()  
  
app.add\_middleware(  
 CORSMiddleware,  
 allow\_origins=["\*"],  
 allow\_credentials=True,  
 allow\_methods=["\*"],  
 allow\_headers=["\*"],  
)  
  
@app.get("/")  
def read\_root():  
 return {"message": "Hello from FastAPI!"}

if \_\_name\_\_ == "\_\_main\_\_":

uvicorn.run(app, host="127.0.0.1", port=8000)

### Budowanie EXE backendu

cd backend  
pyinstaller --onefile --name main main.py

Plik dist/main.exe będzie gotowy do użycia.

## KROK 5: ELECTRON

### Konfiguracja main.js

Plik main.js powinien znajdować się w katalogu głównym. Odpowiada za uruchamianie backendu i ładowanie interfejsu z Reacta.

const { app, BrowserWindow } = require('electron');  
const path = require('path');  
const { spawn, exec } = require('child\_process');  
  
let mainWindow;  
let backendProcess;  
const exePath = path.join(\_\_dirname, 'backend/dist/main.exe');  
  
function createWindow() {  
 backendProcess = spawn(exePath);

backendProcess.stdout.on('data', (data) => {   
 console.log(`[BACKEND STDOUT] ${data}`);   
 });  
 backendProcess.stderr.on('data', (data) => {   
 console.error(`[BACKEND ERROR] ${data}`);   
 });  
 backendProcess.on('exit', (code) => {   
 console.error(`[BACKEND EXIT] Kod zakończenia: ${code}`);   
 });  
  
 mainWindow = new BrowserWindow({  
 width: 800,  
 height: 600,  
 webPreferences: {  
 nodeIntegration: true,  
 contextIsolation: false,  
 },  
 });  
  
 mainWindow.loadFile(path.join(\_\_dirname, 'frontend/dist/index.html'));  
  
 mainWindow.on('before-quit', () => {  
 if (backendProcess) {  
 exec(`taskkill /PID ${backendProcess.pid} /F /T`);  
 mainWindow = null;  
 }  
 });  
}  
  
app.on('ready', createWindow);  
app.on('window-all-closed', () => app.quit());

### Konfiguracja package.json (w katalogu głównym)

Plik package.json w katalogu głównym powinien wyglądać tak:

{  
 "name": "my\_app",  
 "private": true,  
 "main": "main.js",  
 "scripts": {  
 "start": "electron ."  
 },  
 "devDependencies": {  
 "electron": "^29.0.0"  
 }  
}

## KROK 6: Uruchamianie aplikacji

### Uruchamianie aplikacji

cd ..  
npm run start